

Sequence analysis

COBALT: constraint-based alignment tool for multiple protein sequences

Jason S. Papadopoulos and Richa Agarwala*

National Center for Biotechnology Information, National Institutes of Health, Department of Health and Human Services, Bldg. 38A, Room 10–03N, 8600 Rockville Pike, Bethesda, MD 20894, USA

Received on September 15, 2006; revised on January 23, 2007; accepted on February 26, 2007

Advance Access publication March 1, 2007

Associate Editor: Charlie Hodgman

ABSTRACT

Motivation: A tool that simultaneously aligns multiple protein sequences, automatically utilizes information about protein domains, and has a good compromise between speed and accuracy will have practical advantages over current tools.

Results: We describe COBALT, a constraint based alignment tool that implements a general framework for multiple alignment of protein sequences. COBALT finds a collection of pairwise constraints derived from database searches, sequence similarity and user input, combines these pairwise constraints, and then incorporates them into a progressive multiple alignment. We show that using constraints derived from the conserved domain database (CDD) and PROSITE protein-motif database improves COBALT's alignment quality. We also show that COBALT has reasonable runtime performance and alignment accuracy comparable to or exceeding that of other tools for a broad range of problems.

Availability: COBALT is included in the NCBI C++ toolkit. A Linux executable for COBALT, and CDD and PROSITE data used is available at: <http://ftp.ncbi.nlm.nih.gov/pub/agarwala/cobalt>

Contact: richa@helix.nih.gov

1 INTRODUCTION

The simultaneous alignment of multiple sequences (*multiple alignment*) serves as a building block in several fields of computational biology (Gotoh, 1999), such as phylogenetic studies (Fleissner *et al.*, 2005), detection of conserved motifs (Frith *et al.*, 2004), prediction of functional residues and secondary structure (Livingstone and Barton, 1996), prediction of correlations (Socolich *et al.*, 2005) and even quality assessment of protein sequences (Bianchetti *et al.*, 2005). The development of algorithms that can automatically produce biologically plausible multiple alignments is a subject of very active research (Edgar and Batzoglou, 2006; Notredame, 2002). Unfortunately, finding a multiple alignment that rigorously optimizes the commonly used 'sum-of-pairs' scoring measure is computationally hard (Wang and Jiang, 1994) and not practical when more than a few sequences are involved (Li *et al.*, 2000). This has led to an arsenal of approximation techniques from

graph theory (Gupta *et al.*, 1995; Kobayashi and Imai, 1998) combinatorial optimization (Notredame and Higgins, 1996; Zhong, 2002) and probability theory (Do *et al.*, 2005).

Most of the popular modern algorithms designed for multiple alignment of more than a few sequences, such as ClustalW (Thompson *et al.*, 1994), MUSCLE (Edgar, 2004a,b), ProbCons (Do *et al.*, 2005) and PCMA (Pei *et al.*, 2003), employ a progressive alignment technique (Feng and Doolittle, 1987) that aligns pairs of sequences, and then pairs of sequence collections, starting from the most similar sequences and continuing until all sequences contribute to the alignment. These algorithms detect sequence similarity as a preliminary step and use the results to construct a guide tree that drives the actual alignment process. Once an initial solution containing all sequences is available, a refinement stage (Wallace *et al.*, 2005) attempts to use the extra information embodied in the alignment to improve that solution.

Several algorithms take the set of sequences to align as their only input, whereas others incorporate information from multiple heterogeneous sources (Notredame *et al.*, 2000). Even the latter primarily restrict themselves to observations of the input dataset, for example, the secondary structure locations on the inputs (O'Sullivan *et al.*, 2004). When the number of sequences is small or the collection has low pairwise similarity, less information is available for these algorithms to construct an alignment. The information content can be increased by turning sequences into position-specific profiles based on the similarity of each sequence to members of a database, and then aligning the profiles instead of the original sequences (Simossis and Heringa, 2004). Alignment to a profile is significantly more sensitive to subtle relationships between sequences (Gribskov *et al.*, 1987; Marti-Renom *et al.*, 2004). The traditional drawback to use of profiles has been the computational expense of constructing them, for example, via iterated PSI-BLAST searches against a large protein database.

We see three relatively underexplored opportunities for further development in the field of multiple alignment:

- (1) The use of biologically relevant information encoded in databases such as the conserved domain database (Marchler-Bauer *et al.*, 2005) (CDD) and PROSITE (Hulo *et al.*, 2006) for improving the quality of multiple alignments. CDD is a curated collection of

*To whom correspondence should be addressed.

profiles derived from aligned protein families, whereas PROSITE is a database of regular expressions representing motifs. Using independent, curated, biologically significant databases has the additional advantage of potentially improving the alignment quality automatically as and when these databases are updated to represent a larger number of protein families or motifs. PROSITE patterns were used by Du and Lin (2005) to constrain multiple alignments, but we are not aware of any multiple alignment tool that has attempted to utilize CDD.

- (2) The use of local pairwise similarity present in *multiple* sequence pairs to highlight similar regions in otherwise divergent sequences. Local alignments can also constrain global alignment to improve performance, because the presence of a constraint reduces the size of the space that a dynamic programming implementation must search for an optimal pairwise alignment. Some algorithms, such as T-Coffee (Notredame *et al.*, 2000) and DbClustal (Thompson *et al.*, 2000), do use libraries of pairwise alignments, but they do not attempt to explicitly choose alignments present in multiple pairs.
- (3) An easy way for users to specify regions they want to see aligned in any multiple alignment computed. User input can be particularly useful when user knowledge is not reflected in sequence similarity. Such a capability was added to a semi-automatic version of DIALIGN (Morgenstern *et al.*, 1998) by Morgenstern *et al.* (2006). Other methods (for example, SALIGN in MODELLER (Marti-Renom *et al.*, 2004)) include the option for a user to specify constraints when aligning sequences and structures.

We explore these three areas with COBALT (constraint-based alignment tool), a new multiple alignment algorithm for protein sequences.

COBALT has a general framework that uses progressive multiple alignment to combine pairwise constraints from different sources into a multiple alignment. COBALT does not attempt to use all available constraints (for example, via algorithms used by Myers *et al.* (1996)) but uses only a high-scoring *consistent* subset that can change as the alignment progresses, where a set of constraints is called consistent if all of the constraints in the set can be simultaneously satisfied by a multiple alignment. Using the RPS-BLAST tool (the core search algorithm for the service described by Marchler-Bauer and Bryant (2004)), we can quickly search for domains in CDD that match to regions of input sequences. When the same domain matches to multiple sequences, we can infer several potential pairwise constraints based on these domain matches. Furthermore, CDD also contains auxiliary information that allows COBALT to create partial profiles for input sequences before progressive alignment begins, and this avoids computationally expensive procedures for building profiles. We use PROSITE patterns for making constraints only in the refinement stage since using them in the initial stages gave worse performance (data not shown). This is likely because PROSITE patterns are much shorter than domains from CDD

and as such are more likely to give spurious matches. COBALT also retains a maximum consistent subset of any user-specified pairwise constraints by giving these constraints a priority higher than that for any constraint derived by other means.

We tested COBALT on five different multiple alignment benchmark sets. Compared with ClustalW, MUSCLE, ProbCons and PCMA, COBALT achieves the highest or close to highest average alignment quality, although all five programs perform similarly on many of these benchmarks. Here, the figure of merit is the percentage of letter pairs in computed alignments that match those in the conserved regions of reference alignments. Use of CDD searches improves COBALT's average alignment quality by $\sim 3\%$, and use of local alignments significantly improves alignment quality in benchmarks such as Implanted Rose Motifs base (IRMbase) (see Table 1). We also show that the alignments reported by various alignment algorithms differ significantly, and this is an important consideration when making conclusions based on multiple alignment produced by any tool (Ogden and Rosenberg, 2006).

The runtime performance of COBALT is highly data driven, but we find empirically that our implementation is about two times slower than MUSCLE and comparable to ProbCons and PCMA unless the number of sequences exceeds about a dozen, in which case COBALT is about five times faster than ProbCons. COBALT, therefore, represents a good compromise between alignment quality and runtime requirements and may be a good choice when one does not want to try multiple tools. We expect to incorporate COBALT into various NCBI resources and make further enhancements to improve COBALT's speed and/or accuracy.

In the next section, we describe the methods used by COBALT to find constraints and generate a guide tree, along with the heuristics for aligning subsets of sequences presented by the guide tree. This section also describes five benchmarks used for evaluating COBALT. The Results section compares alignment quality achieved by COBALT with that of ClustalW, MUSCLE, ProbCons and PCMA on the five benchmarks. We close with a discussion of related work and open problems.

2 MATERIALS AND METHODS

COBALT is included in the NCBI C++ Toolkit. Numerous auxiliary programs were written in C, C++ and Perl to automate testing and summarize results. Splus version 6.0 was used to run Friedman's rank sum test.

2.1 Scoring a multiple alignment

The traditional method of scoring an alignment between a pair of sequences is to use a score matrix based on log-odds scores, such as the PAM or BLOSUM series of matrices. When there are more than a few sequences, the information content in a multiple alignment can be measured by its entropy. A small number of sequences, or correlations between residues, can have a harmful effect on an entropy-based scoring measure. Partitioning residues into a smaller number of residue classes can sometimes dampen this effect.

The intuition used by Tharakaraman *et al.* (2005) is to combine ideas behind log-odds and entropy-based scoring. This method scores a

multiple alignment by measuring how well individual sequences agree with the overall alignment. We extend these ideas to account for gaps in multiple alignments. We also partition the residues into ten classes when scoring a multiple alignment. The score assigned to a column i , denoted by $CS(i)$, within a multiple alignment is given by:

$$\sum_{j|c_{ij}>0} \frac{c_{ij}}{N} \left[\log \left[\frac{N-1}{K} + \frac{c_{ij}-1}{p_j} \right] - \log \left[\frac{(N-1)(K+A)}{K} \right] \right] \quad (1)$$

where c_{ij} is the number of sequences whose residue in column i of the multiple alignment belongs to class j , N is the number of sequences in the multiple alignment, p_j is the sum of background residue frequencies for residues in class j , A is $\sum_j p_j$ for classes with non-zero c_{ij} and K is a pseudocount set to two by default. The score for a multiple alignment is the sum of the scores for all of its columns. Equation (1) was derived using ideas from Tharakaraman *et al.* (2005) as follows.

The ‘individual score’ in Equation (7) from Tharakaraman *et al.* (2005) is the measure of how similar a residue j in a given sequence S is to column i of a given profile:

$$s_{ij} = \log \left[\left(\frac{c_{ij} + a_j}{c + a} \right) / p_j \right]$$

This equation essentially gives the log-odds score of placing residue j in column i and is also used by PSI-BLAST to calculate expect values. However, to our knowledge, all applications that use the above equation ignore columns containing gaps.

For scoring multiple alignments, we can use this formulation by checking how each sequence S scores against the alignment for the rest of the sequences. Because c_{ij} includes the count for S in the multiple alignment for S , the count for c_{ij} should be decremented by one while scoring S . Also, since columns with gaps will not have $\sum_j c_{ij} = N$, we explicitly replace c by $N-1$ so as to downweight the score for columns with gaps. These two substitutions give:

$$s_{ij} = \log \left[\left(\frac{(c_{ij}-1) + a_j}{(N-1) + a} \right) / p_j \right]$$

Because there are c_{ij} sequences contributing residue j to column i , the overall average contribution of residue j to score for column i is:

$$\frac{c_{ij}}{N} \log \left[\left(\frac{(c_{ij}-1) + a_j}{(N-1) + a} \right) / p_j \right]$$

for a total score of

$$CS(i) = \sum_{j|c_{ij}>0} \frac{c_{ij}}{N} \log \left[\left(\frac{(c_{ij}-1) + a_j}{(N-1) + a} \right) / p_j \right] \quad (2)$$

Next, we use pseudocounts that depend on the background frequencies of residues. Specifically, we set $a_j = p_j * (N-1)/K$ where K is a fixed constant. This gives:

$$a = \frac{N-1}{K} \sum_{j|c_{ij}>0} p_j$$

Using $A = \{\sum_{j|c_{ij}>0} p_j\}$, and substituting values for a_j and a , Equation (2) becomes:

$$CS(i) = \sum_{j|c_{ij}>0} \frac{c_{ij}}{N} \log \left[\left(\frac{(c_{ij}-1) + p_j \frac{N-1}{K}}{(N-1) + \frac{N-1}{K} A} \right) / p_j \right]$$

which is same as Equation (1).

2.2 Algorithm

COBALT is a flexible tool for simultaneously aligning a given set of protein sequences, where users can directly specify pairwise constraints and/or ask COBALT to generate the constraints using sequence

similarity, (optional) CDD searches and (optional) PROSITE pattern searches. COBALT will optionally create partial profiles for input sequences based on any CDD search results. Aside from these features, the COBALT algorithm is similar to that of other progressive multiple alignment tools:

Step 1: Find alignments for generating constraints.

Step 2: Find partial profiles and a pairwise consistent set of constraints.

Step 3: Generate a guide tree.

Step 4: Create a multiple alignment using the current set of constraints and guide tree.

Step 5: Create bipartitions and realign.

Step 6: Perform (optional) refinement by determining a new set of constraints and iterating from Step 4 as long as the number of constraints keeps increasing.

We do not regenerate the guide tree in the refinement phase, because we found that the guide tree generated in Step 3 provides a branching order for progressive alignment that can lead to the desired benchmark solution, and do not expect that regenerating the tree will improve result quality. Next, we briefly describe the implementation of the above steps. In the following, we denote the given set of input protein sequences by $S = \{S_1, S_2, \dots, S_N\}$, the residues of sequence S_i by r_i^1, \dots, r_i^m where m is the length of S_i , and the profile for S_i at position j by $f_i^j[1 \dots k]$ for the protein alphabet of size k . We use $f_i^j[0]$ to represent the frequency of gaps for S_i at position j . From now on, we overload the term *residue* to mean an index in a scoring matrix (BLOSUM62 by default), and also the actual amino acid letter in the sequence.

2.2.1 Finding alignments for generating constraints

(Step 1) RPS-BLAST is used to align each S_i to each domain in the CDD database. For each domain, CDD contains a position-specific score matrix used for RPS-BLAST alignment, the residue frequencies that produced the score matrix, and a list of both highly conserved (*core block*) and divergent (*loop*) regions within the domain. For each S_i , we divide each domain match that meets a minimum expect-value threshold (0.01 by default) into a collection of core blocks, and realign each individual core block to S_i using dynamic programming and the portion of the domain’s score matrix appropriate for the block. During realignment, a block may shift position from its location on the original match up to half the size of the loop region to either side of the block, and may have gaps added or removed compared to the original match. Because sequences can be expected to align on block boundaries, we use only the block alignments for inferring a potential constraint between S_i and S_j , and only if both of them align to the same portion of a domain.

The BLASTP module of BLAST is used after RPS-BLAST to find local pairwise sequence similarity in regions where RPS-BLAST fails to detect possible structural similarities to CDD domains. Each sequence S_i is partitioned into regions that participate in a constraint based on CDD alignments (*domain regions*) and regions that do not (*filler regions*). Filler regions of S_i are aligned to set $S - \{S_i\}$ using BLASTP, and any match found that exceeds a minimum expect-value (0.01 by default) becomes a potential pairwise constraint.

Matches against protein motifs from the PROSITE database are found using PHI-BLAST (Zhang *et al.*, 1998). Each occurrence of a pattern on S_i makes a potential pairwise constraint with each occurrence of the same pattern on S_j when $i \neq j$.

The initial set of potential constraints is the set of CDD alignments, pairwise local alignments and any user-specified residue pairs between sequences. Matches against protein motifs are used only in the refinement stage.

2.2.2 Consistent constraints and profiles (Step 2) Pairwise constraints generated in Step 1 may conflict with each other. For each pair of sequences with constraints, we find a consistent subset by determining the maximum-scoring collection of pairwise constraints between the pair of sequences, such that the sequence ranges that appear in all constraints are disjoint. Here, the score for a constraint derived from BLASTP or RPS-BLAST is the alignment score; user-defined constraints are all given an artificially high score to preserve the maximum number of user-specified constraints.

If S_i aligns to one or more domains in CDD, then we use the position-specific matrix of residue frequencies for those domains to create a partial profile for S_i in the locations dictated by block alignments. The domain matches that will contribute residue frequencies are chosen in a greedy manner, where the next domain chosen does not overlap any domain matches chosen for S_i so far, and has the highest-scoring set of block alignments. Two alignments are said to overlap in this context, if their range on S_i overlaps. We add only residue frequencies from block alignment regions, boosting the frequency of the actual letter in each position of S_i as follows:

$$f_i^j[k] = \begin{cases} 1 & \text{if } k = r_i^j \text{ and } j \notin B \\ 0 & \text{if } k \neq r_i^j \text{ and } j \notin B \\ d + (1-d) * b_i^j[k] & \text{if } k = r_i^j \text{ and } j \in B \\ (1-d) * b_i^j[k] & \text{if } k \neq r_i^j \text{ and } j \in B \end{cases}$$

where, domain i' contributes its column j' to position j of S_i , $b_i^j[k]$ is the frequency of residue k in column j' of domain i' , d is a parameter in the range $(0, 1)$ used to upweight the frequency of the actual residue at position j of S_i and B is a chosen block alignment. The value of d is user specified and defaults to 0.5. Although the resulting profiles likely will not completely cover the input sequences, they do increase the total information content. If S_i has no matches to any domain in CDD, then the profile for S_i simply reflects the residue at each position.

2.2.3 Finding a guide tree (Step 3) The pairwise consistent constraints found in Step 2 are used to generate a distance matrix. Let the score between sequence S_i and S_j be score_{ij} , corresponding to the combined alignment score of all constraints found in Step 2 for this pair of sequences. The distance between two sequences S_i and S_j is

$$d_{ij} = 1 - \left(\frac{\text{score}_{ij}}{2} \right) * \left(\frac{1}{\text{self}_i} + \frac{1}{\text{self}_j} \right)$$

where self_k is the BLOSUM62 score of aligning S_k to itself. This distance formula is a slight modification of the metric used by Clarke *et al.* (2002) and can be interpreted as the average identity between S_i and S_j . We have found this metric to be robust against disparities in sequence length while incorporating score matrix information in a natural way. Pairwise distances become an alignment guide tree via neighbor joining (Hillis *et al.*, 1996; Saitou and Nei, 1987). The guide tree can optionally be constructed using fast minimum evolution (FASTme) (Desper and Gascuel, 2002), but we did not do so in the results presented here. We note that the results produced using FASTme were generally comparable to the ones presented here.

2.2.4 Computing the multiple sequence alignment (Step 4) The progressive multiple alignment does a depth-first traversal of the tree generated in Step 3. At each node of the tree, we generate profiles for both subtrees and align these profiles to produce a multiple alignment for all of the sequences seen thus far.

The profile of a subtree is computed by adding the contribution from each sequence in the subtree. For the subtree containing S_i , the contribution of sequence S_i to position k of column j in the profile is $w_i * f_i^j[k]$, where w_i is the weight for S_i , computed by normalizing the reciprocals of the distance from each sequence to the subtree root. In practice, this formulation reduces the contribution of more distantly

related sequences but tends to produce equal weights for all sequences when the tree is ambiguous.

A variant of ordinary Needleman–Wunsch dynamic programming computes a global alignment of two profiles. The alignment process uses well-known techniques to reduce memory consumption (Edgar, 2000a) and includes two variations that are specific to profile alignments. The first variation is the choice of profile–profile score function (Edgar and Sjölander, 2004; Wang and Dunbrack, 2004). Given an amino acid score matrix M and two vectors p and q of residue frequencies, where vector element i is the frequency of occurrence of amino acid i , a straightforward generalization for the pairwise score M_{ij} for aligning amino acid i with amino acid j is

$$\sum_{i=1}^k \sum_{j=1}^k p[i] * q[j] * M_{ij}$$

However, this scoring function has the effect of diluting the profile–profile score because of the influence of ‘cross-terms’ even when two profiles are exact copies of each other. A more appropriate scoring function should assign a high score to pairs of profile columns that contain similar residue frequency distributions. This suggests a modification to the above formula:

$$\sum_{i=1}^k g_i * M_{ii} + \frac{\sum_{i=1}^k \sum_{j=1}^k (p[i] - g_i) * (q[j] - g_j) * M_{ij}}{1 - \sum_{i=1}^k g_i} \quad (3)$$

where $g_i = \min(p[i], q[i])$. The first summation is the ‘common score’ for the two profile columns, whereas the second is the contribution of the cross terms, normalized to preserve the scaling of S . Since cross terms contribute to the score for only those pairs i and j where $p[i] > q[i]$ and $p[j] < q[j]$ (so that $g_i = q[i]$ and $g_j = p[j]$), their effect is restricted to only the ‘discrepant portions’ of the two profiles. We ensure that the second term is computed only if $\sum_{i=1}^k g_i \neq 1$. We found that limiting the influence of cross terms improves alignment quality, and because at least half of the cross terms are zero, the inner loop of the dynamic programming can avoid many needless computations.

The second variation involves modifying the Needleman–Wunsch recurrence relations to account for the frequency of gaps in vector p . Affine gap scoring means that the first gap character incurs a gap-open penalty G_o , while remaining consecutive gaps only incur a gap-extension penalty G_e . When opening a gap in vector p , we scale the penalty by the fraction of non-gap characters in q ; i.e. the gap open penalty is:

$$G_o * (1 - q[0]) \text{ or } G_o * (1 - p[0]) \quad (4)$$

depending on whether the gap is opened in p or q , respectively. Similarly, when aligning columns p and q , we add the gap extension penalty

$$G_e * (p[0] * (1 - q[0]) + q[0] * (1 - p[0])) \quad (5)$$

to the profile–profile score. These modifications reduce the cost of aligning profile regions that contain many gaps, and further improve alignment quality. Equations (3–5) represent all of the changes to the dynamic programming recurrence relations needed to align profiles instead of sequences.

Each alignment of two sequence collections may also incorporate pairwise constraints that reduce the size of the dynamic programming lattice to explore. We tabulate constraints that cross from one subtree to the other, and the highest-scoring consistent subset constrains the dynamic programming procedure. We also merge identical constraints in the same profile neighborhood, scaling the constraint score by the number of constraints merged. The merge process makes it more likely that constraints appropriate to *multiple* sequences will influence the complete profile–profile alignment.

2.2.5 Alignments by bipartition (Step 5) The initial guide tree is rooted at its longest edge. In an attempt to undo errors committed by this arbitrary choice for the root, we isolate a fraction of the longest edges in the guide tree as possible choices for the root. For each choice, we partition the current multiple alignment into two sequence collections separated by that edge and then perform a profile–profile alignment between the two collections. Each column in the realignment is scored using Equation 1. For each edge, bipartition repeats up to five times, or as long as the best score from the current iteration improves the best score from the previous iteration by at least 2%.

2.2.6 Refinement by finding new constraints (Step 6)

The second refinement phase begins by finding conserved columns in the output from Step 5. A column in a multiple alignment is considered to be high scoring if its score exceeds a cutoff (set to 0.67 by default), and groups of at least two adjacent high-scoring columns are considered conserved. Iteration continues as long as the number of conserved columns increases. Before iterating, the set of constraints found in Step 2 is replaced by constraints that encompass alignment decisions based on conserved columns, pattern matches and user-specified pairs.

COBALT uses an all-against-all collection of pairwise constraints to represent each group of conserved columns. Conserved columns may contain gaps, but sequences that contain gaps in a conserved column do not participate in pairwise constraints for that column. This exception allows conserved columns to be used for most profile–profile alignments, while generating pressure on slightly misaligned sequences to shift position.

2.3 Assessment

Most of the development of COBALT was done using BaliBase 2.0 (Bahr *et al.*, 2001), containing 265 alignments divided into eight sets according to sequence length and percent similarity. These sets represent a wide variety of multiple alignment problems. We used the following benchmarks for our tests:

- (1) **BaliBase 3.0** (Thompson *et al.*, 2005), containing 218 alignments organized in the same way as BaliBase 2.0, but with larger sequence collections that contain more outlier sequences.

Table 1. Q-score for COBALT, ClustalW, MUSCLE, PCMA and ProbCons restricted to core regions on various benchmarks

Tool	Benchmark					Running time
	Bali	HOM	IRM	PREFAB	SAB	
ProbCons	86.41	82.03	83.92	71.64	49.59	61 h 31 min
PCMA	85.75	80.37	90.01	69.76	46.27	11 h 57 min
MUSCLE	82.35	80.92	43.22	67.81	45.38	2 h 22 min
ClustalW	75.37	80.23	13.62	61.70	43.56	2 h 25 min
COBALT	84.44	84.40	88.13	67.05	50.50	8 h 54 min
COBALT without some additional information						
No patterns	84.36	84.40	88.01	67.01	50.42	8 h 48 min
No freq	82.29	81.65	88.13	65.15	46.63	8 h 48 min
No RPS	81.45	80.03	88.13	64.45	44.30	4 h 19 min
No info.	81.52	80.02	88.01	64.57	44.18	4 h 14 min

Highest Q-score for each benchmark is shown in bold. Rows labeled ‘No patterns’, ‘No freq’, ‘No RPS’, and ‘No info.’ show results when COBALT is not constrained by PROSITE patterns, residue frequencies from CDD, any information from CDD and any PROSITE pattern or CDD, respectively.

- (2) **HOMSTRAD** (Stebbins and Mizuguchi, 2004), containing 1032 alignments that represent a large series of known protein families.
- (3) **IRMbase** (Stoye *et al.*, 1998), containing 180 alignments. In this set, a highly conserved motif is inserted into large, randomly generated protein sequences, and then edit operations are performed that simulate evolutionary events on the collection of sequences. The objective is to recover the conserved motif.
- (4) **PREFAB 4.0** (Edgar, 2004a), containing 1682 alignments that consist of two sequences surrounded by a collection of other similar sequences found by PSI-BLAST. The objective is to produce a multiple alignment that contains the known structural alignment of the original pair of sequences.
- (5) **SABmark** (Walle *et al.*, 2005) has 634 sets of sequence pairs, and the objective is to produce a multiple alignment that simultaneously preserves the known structural alignments of all pairs of sequences in each dataset.

Using these benchmarks, we compared COBALT to ClustalW 1.83, MUSCLE 3.6, ProbCons 1.10 and PCMA 2.0. Default settings were used for all programs except for the results presented for COBALT without some or any additional information. CDD version 2.05 and PROSITE release 19.0 were used for the COBALT results reported here. The quality assessment score (*Q-score*) is an average over all datasets in a benchmark, where for each dataset we find the percentage of the letter pairs in the reference alignment that are also aligned in the computed alignment. BaliBase, PREFAB and IRMbase benchmarks mark core regions in their reference alignments; for these benchmarks, we also calculate the *Q-score* for letter pairs in only the core regions while considering the whole alignment as a core region for HOMSTRAD and SABmark.

3 RESULTS

COBALT was developed using BaliBase 2.0 and tested on BaliBase 3.0, HOMSTRAD, PREFAB, IRMbase and SABmark multiple alignment benchmarks. Table 1 shows the *Q-score* for alignments computed by ClustalW, MUSCLE, ProbCons, PCMA and COBALT on five reference benchmarks and their running time. The *Q-score* restricted to core regions gives an indication of how well each algorithm finds these regions. These results show that COBALT achieves the best score for HOMSTRAD and SABmark. COBALT also achieves a score comparable to the best score on BaliBase 3.0 (achieved by ProbCons), PREFAB (achieved by ProbCons) and IRMbase (achieved by PCMA). Table 1 also shows that COBALT is significantly faster than ProbCons. The results in Table 1 for IRMbase show that the isolated nature of all conserved regions defeats the similarity-detecting heuristics in, MUSCLE and ClustalW. The datasets from each benchmark, where COBALT performs the best compared to all algorithms are bali_20002 (94.3 versus best of 89.4 by PCMA), hom_SpoU_methylase_N (97.2 versus best of 43.1 by ProbCons), irm_1_400_4_30_7 (100.00 versus best of 48.9 by PCMA), 1h9jA_1g291 (63.2 versus best of 21.6 by MUSCLE) and twi_156 (84.3 versus best of 22.7 by ProbCons).

The average *Q-score* for a benchmark hides the fact that there is usually significant variation on any given set. We quantify the variation in each benchmark, reported in Table 2, by finding the root mean square deviation in *Q-scores* for a pair of tools, and then finding the significance in the difference in results using Friedman’s rank sum test. We note that although

Table 2. Root mean square deviation of Q-score for COBALT, ClustalW, MUSCLE, PCMA and ProbCons restricted to core regions on various benchmarks

Tool	Bali	HOM	IRM	PREFAB	SAB
COBALT versus ProbCons	6.71 (–s)	9.34 (s)	16.61 (0.8175)	16.39 (–s)	9.98 (0.1208)
COBALT versus PCMA	8.76 (–0.001)	11.60 (s)	15.27 (–s)	17.35 (–s)	12.99 (s)
COBALT versus MUSCLE	10.32 (0.0094)	10.07 (s)	51.66 (s)	16.48 (–0.0011)	13.05 (s)
COBALT versus ClustalW	16.88 (s)	12.37 (s)	76.92 (s)	20.89 (s)	14.51 (s)
ProbCons versus PCMA	7.02 (0.0011)	8.17 (s)	17.56 (–0.0017)	13.23 (s)	9.01 (s)
ProbCons versus MUSCLE	9.69 (s)	6.22 (s)	46.68 (s)	13.49 (s)	9.91 (s)
ProbCons versus ClustalW	17.31 (s)	9.75 (0.0004)	73.04 (s)	20.87 (s)	13.43 (s)
PCMA versus MUSCLE	10.97 (s)	7.94 (0.4581)	54.01 (s)	14.90 (0.0427)	9.46 (0.09)
PCMA versus ClustalW	17.32 (s)	5.93 (–0.1797)	79.14 (s)	18.97 (s)	11.77 (s)
MUSCLE versus ClustalW	13.21 (s)	8.64 (–0.1933)	36.73 (s)	18.36 (s)	10.41 (0.0059)

Significance is given in brackets and is calculated using Friedman's rank sum test, where the value 's' means a P -value of $< 1E-10$. A negative P -value means that the method on the right performed better (had a lower average rank) than the method on the left.

the average performance of all four programs is quite similar on HOMSTRAD, the tools show large variations in alignment quality for any particular dataset. Because of this variation, we think that users should consider using more than one tool, but if users want to pick one tool, then COBALT provides a good balance between alignment quality and running time.

As shown in Table 1, using CDD improves the Q-score for COBALT by a few percentage points, whereas PROSITE patterns have a negligible effect on the results. This shows that there are gains to be made by utilizing resources containing information about protein domains, and also shows that the algorithm used by COBALT to make an alignment, even without using any additional information (Table 1, row labeled 'No info.'), is comparable to that of current state-of-the-art multiple alignment algorithms.

4 DISCUSSION

COBALT is a general framework for transforming pairwise constraints among multiple protein sequences into a multiple sequence alignment. The constraints may arise from several unrelated sources, and in particular may include constraints derived from direct user input. We believe that by making COBALT more aware of what is already known about proteins and captured in publicly available resources, COBALT has a better chance of producing a biologically meaningful multiple alignment compared to tools that do not utilize this information. The alignment process itself includes several heuristics for combining constraints and aligning sequences represented as frequency profiles. The result is an algorithm whose performance matches or exceeds that of the best current methods and still achieves reasonable running time.

Our current implementation uses searches against the PROSITE database of protein-motif regular expressions and against the CDD of protein domains. We expect COBALT alignment quality to improve as the underlying resources continue to evolve. Future efforts will investigate the applicability of additional information such as secondary structure alignments computed with recent algorithms (Shindyalov and Bourne, 1998; Zhou and Zhou, 2005) and the detection of short

highly conserved motifs found with *de novo* methods (Neuwald *et al.*, 1997; Rigoutsos and Floratos, 1998). We are particularly interested in finding robust and computationally inexpensive motif-finding tools, as we find that PROSITE patterns longer than three letters are highly selective: performing the pattern search procedure on the datasets comprising BaliBase 2.0 shows that over 90% of the resulting constraint positions agree exactly with the reference alignment.

Progressive multiple alignment algorithms all have difficulty with highly divergent sequence inputs, and so COBALT may also benefit from incorporating alignment algorithms that explicitly process more than two sequences or sequence collections at a time (Kececioglu and Starrett, 2004; Schroedl, 2005; Zhang and Kahveci, 2006). Unfortunately, preliminary investigations show that these measures invariably require excessive computational resources.

We are also examining ways to improve performance in the case where the similarity between input sequences is high. It is especially desirable to avoid the need for RPS-BLAST against CDD if there is no need to deduce subtle structural relationships between inputs, and COBALT should be able to detect this situation and avoid unnecessary work that accounts for a large portion of the algorithm's runtime. Because RPS-BLAST easily runs in parallel, we are also considering parallelizing at least some computations in COBALT as part of continuing development and optimization.

Finally, we have implemented COBALT as a library in the NCBI C++ Toolkit and expect to incorporate the algorithm into NCBI resources and into user tools such as alignment editors.

ACKNOWLEDGEMENTS

Thanks to David Lipman, Jim Ostell and Tom Madden for advice and encouragement. Discussions with John Spouge, Teresa Przytycka, Maricel Kann, Anna Panchenko and Aron Marchler-Bauer were also helpful. A web page developed by Irena Zaretskaya to run the COBALT algorithm has been helpful in jump-starting activity on linking COBALT with other applications at NCBI. We thank Aravind Iyer for testing COBALT with his datasets and providing valuable feedback.

This research was supported by the Intramural Research Program of the NIH, NLM. Funding to pay the Open Access charges was provided by the Intramural Research program of the NIH, National Library of Medicine.

Conflict of Interest: none declared.

REFERENCES

- Bahr, A. *et al.* (2001) BALiBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations. *Nucleic Acids Res.*, **29**, 323–326.
- Bianchetti, L. *et al.* (2005) vALiD: validation of protein sequence quality based on multiple alignment data. *J. Bio. Comput. Biol.*, **3**, 929–947.
- Clarke, G.D.P. *et al.* (2002) Inferring genome trees by using a filter to eliminate phylogenetically discordant sequences and a distance matrix based on mean normalized BLASTP scores. *J. Bacteriol.*, **184**, 2072–2080.
- Desper, R. and Gascuel, O. (2002) Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *J. Comput. Biol.*, **9**, 687–705.
- Do, C.B. *et al.* (2005) ProbCons: probabilistic consistency-based multiple sequence alignment. *Genome Res.*, **15**, 330–340.
- Du, Z. and Lin, F. (2005) Pattern-constrained multiple polypeptide sequence alignment. *Comput. Biol. Chem.*, **29**, 303–307.
- Edgar, R.C. (2004a) MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, **5**, 113.
- Edgar, R.C. (2004b) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, **32**, 1792–1797.
- Edgar, R. C. and Batzoglou, S. (2006) Multiple sequence alignment. *Curre. Opin. Struct. Biol.*, **16**, 368–373.
- Edgar, R. C. and Sjölander, K. (2004) A comparison of scoring functions for protein sequence profile alignment. *Bioinformatics*, **20**, 1301–1308.
- Feng, D.-F. and Doolittle, R.F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, **25**, 351–360.
- Fleissner, R. *et al.* (2005) Simultaneous statistical multiple alignment and phylogeny reconstruction. *Sys. Biol.*, **54**, 548–561.
- Frith, M. C. *et al.* (2004) Finding functional sequence elements by multiple local alignment. *Nucleic Acids Res.*, **32**, 189–200.
- Gotoh, O. (1999) Multiple sequence alignment: algorithms and applications. *Adv. Biophys.*, **36**, 159–206.
- Gribskov, M. *et al.* (1987) Profile analysis: Detection of distantly related proteins. *Proc. Nat. Acad. Sci. USA*, **84**, 4355–4358.
- Gupta, S. K. *et al.* (1995) Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *J. Comput. Biol.*, **2**, 459–472.
- Hillis, D.M. *et al.* (1996) *Molecular Systematic*. Sinauer Associates, Sunderland, M, second edition.
- Hulo, N. *et al.* (2006) The PROSITE database. *Nucleic Acids Res.*, **34**, D227–D230.
- Kececioglu, J.D. and Starrett, D. (2004) Aligning alignments exactly. In *Proceedings of the 8th ACM Conference Research in Computational Molecular Biology*, pp. 85–96.
- Kobayashi, H. and Imai, H. (1998) Improvement of the A* algorithm for multiple sequence alignment. In *Proceedings of the Genome Informatics Workshop*, **9**, 120–130.
- Li, M. *et al.* (2000) Near optimal multiple alignment within a band in polynomial time. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, pp. 425–434.
- Livingstone, C.D. and Barton, G.J. (1996) Identification of functional residues and secondary structure from protein multiple sequence alignment. *Meth. Enzymol.*, **266**, 497–512.
- Marchler-Bauer, A. and Bryant, S.H. (2004) CD-Search: protein domain annotations on the fly. *Nucleic Acids Res.*, **32**, W327–W331.
- Marchler-Bauer, A. *et al.* (2005) CDD: A conserved domain database for protein classification. *Nucl. Acids Res.*, **33**, D192–D196.
- Marti-Renom, M.A. *et al.* (2004) Alignment of protein sequences by their profiles. *Protein Sci.*, **13**, 1071–1087.
- Morgenstern, B. *et al.* (1998) DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics*, **14**, 290–294.
- Morgenstern, B. *et al.* (2006) Multiple sequence alignment with userdefined anchor points. *Algorithms Mol. Biol.*, **1**, <http://www.almob.org/content/1/1/6>
- Myers, G. *et al.* (1996) Progressive multiple alignment with constraints. *J. Comput. Biol.*, **3**, 563–572.
- Neuwald, A. F. *et al.* (1997) Extracting protein alignment models from the sequence database. *Nucleic Acids Res.*, **25**, 1665–1677.
- Notredame, C. (2002) Recent progresses in multiple sequence alignment: a survey. *Pharmacogenomics*, **3**, 131–144.
- Notredame, C. and Higgins, D.G. (1996) SAGA: sequence alignment by genetic algorithm. *Nucleic Acids Res.*, **24**, 1515–1524.
- Notredame, C. *et al.* (2000) T-coffee: a novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, **302**, 205–217.
- Ogden, T.H. and Rosenberg, M.S. (2006) Multiple sequence alignment accuracy and phylogenetic inference. *Systematic Biol.*, **55**, 314–328.
- O Sullivan, O. *et al.* (2004) 3DCoffee: combining protein sequence and structures within multiple sequence alignments. *J. Mol. Biol.*, **340**, 385–395.
- Pei, J. *et al.* (2003) PCMA: fast and accurate multiple sequence alignment based on profile consistency. *Bioinformatics*, **19**, 427–428.
- Rigoutsos, I. and Floratos, A. (1998) Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm. *Bioinformatics*, **14**, 55–67.
- Saitou, N. and Nei, M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. biol. evol.*, **4**, 406–425.
- Schroedl, S. (2005) An improved search algorithm for optimal multiple sequence alignment. *Journal of Artificial Intelligence Research*, **23**, 587–623.
- Shindyalov, I.N. and Bourne, P.E. (1998) Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Eng.*, **11**, 739–747.
- Simossis, V.A. and Heringa, J. (2004) PSI-PRALINE: A novel algorithm for multiple sequence alignment. *Poster in 12th International Conference on Intelligent Systems for Molecular Biology, Glasgow, Scotland*.
- Socolich, M. *et al.* (2005) Evolutionary information for specifying a protein fold. *Nature*, **437**(7058), 512–518.
- Stebbins, L.A. and Mizuguchi, K. (2004) HOMSTRAD: recent developments of the homologous protein structure alignment database. *Nucleic Acids Res.*, **32**, D203–D207.
- Stoye, J. *et al.* (1998) Rose: generating sequence families. *Bioinformatics*, **14**, 157–163.
- Tharakaraman, K. *et al.* (2005) Alignments anchored on genomic landmarks can aid in the identification of regulatory elements. *Bioinformatics*, **21** (Suppl. 1), i440–i448.
- Thompson, *et al.* (1994) CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4680.
- Thompson, J.D. *et al.* (2000) DbClustal: rapid and reliable global multiple alignments of protein sequences detected by database searches. *Nucleic Acids Res.*, **28**, 2919–2926.
- Thompson, J.D. *et al.* (2005) BALiBASE 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins: Structure, Function and Bioinformatics*, **61**, 127–136.
- Wallace, I.M. *et al.* (2005) Evaluation of iterative alignment algorithms for multiple alignment. *Bioinformatics*, **21**, 1408–1414.
- Walle, I.V. *et al.* (2005) SABmark: a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics*, **21**, 1267–1268.
- Wang, G. and Dunbrack, R. L.Jr. (2004) Scoring profile-to-profile sequence alignments. *Protein Science*, **13**, 1612–1626.
- Wang, L. and Jiang, T. (1994) On the complexity of multiple sequence alignment. *J. comput. biol.*, **1**(4), 337–348.
- Zhang, X. and Kahveci, T. (2006) A New Approach for Alignment of multiple proteins. *Pac. Symp. Biocomput.*, **11**, 339–350.
- Zhang, Z. *et al.* (1998) Protein sequence similarity searches using patterns as seeds. *Nucleic Acids Res.*, **26**, 3986–3990.
- Zhong, W. (2002) Using travelling salesman problem algorithms to determine multiple sequence alignment orders. *M.S. Thesis, University of Georgia*.
- Zhou, H. and Zhou, Y. (2005) SPEM: improving multiple sequence alignment with sequence profiles and predicted secondary structures. *Bioinformatics*, **21**, 3615–3621.